Camera Model Design

Voicu Popescu
Purdue University

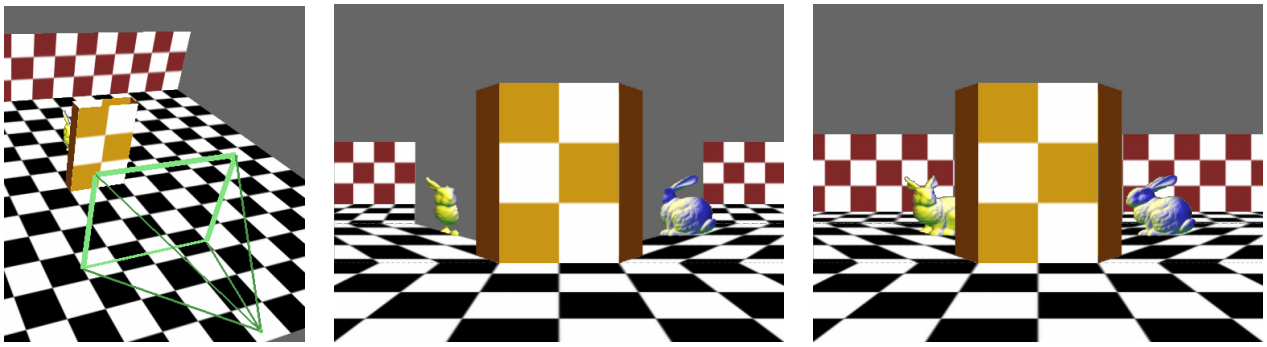

Figure 1 Disocclusion example. The bunny is occluded by the vertical block (*left*). The graph camera (disjoint variant *middle* and overlapping variant *right*) eliminates the occlusion.

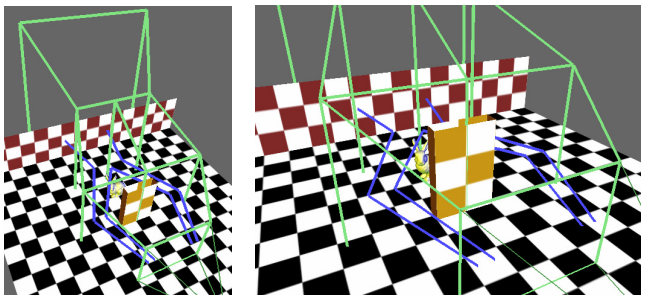

Figure 2 Visualizations of the graph camera used in Figure 1. Green and blue lines show the frustum and four sample rays, respectively.

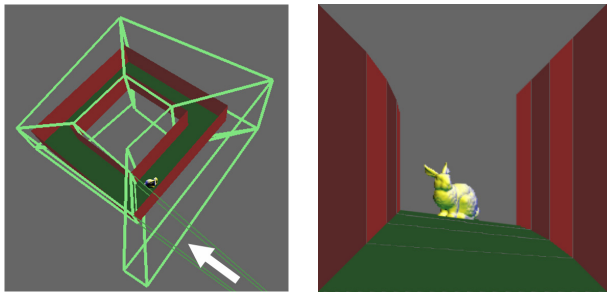

Figure 3 Corridor linearization example. The graph camera view frustum wraps around (*left*), to reveal the entire corridor (*right*).

Abstract

Camera models are essential infrastructure in computer graphics, visualization, and vision. The most frequently used model is the planar pinhole camera, because it approximates the human eye well, producing familiar images, and because it is simple, enabling efficient software and hardware implementations. However, the requirement that all rays pass through a common point is restrictive and relatively little has been done to remove this pinhole constraint. We believe that the apprehension regarding non-pinhole camera models is based largely on misconceptions, such as the belief that these cameras do not produce useful images or that they are inherently inefficient.

In this paper we introduce a novel paradigm for interactive computer graphics based on devising efficient and effective non-pinhole camera models. Instead of using one of the few "off the shelf" cameras, the camera model is designed to meet the application's needs and is dynamically optimized for the data set at hand. We show that powerful NPHC models can be designed with fast projection, which enables efficient feed-forward rendering. To exemplify the proposed paradigm we introduce the graph camera, a malleable but efficient non-pinhole that creates comprehensive images of heavily occluded environments.

CR Categories: I.3.m. [Computer Graphics]: Picture/Image Generation– Viewing algorithms.

Keywords: camera model, interactive 3D computer graphics.

1 Introduction

Camera models define the correspondence between image pixels and captured rays, and are essential infrastructure for many applications in computer graphics, visualization, and computer vision. By far the most frequently used model is the planar pinhole camera (PPHC). One reason is that the PPHC closely approximates the human eye, producing images familiar to us. Another reason is that the PPHC model is simple. Physical implementations exist for over one hundred years. Current cameras are compact, inexpensive, and have resolutions in the millions of pixels. The simplicity of the PPHC model also enables efficient *virtual* implementations. Inexpensive graphics hardware computes at interactive rates PPHC images of scenes modeled with millions of triangles.

However, the PPHC model is restrictive. While the field of view limitation has been addressed by innovations such as fisheye lenses, panning cameras, and omnidirectional catadioptric cameras, relatively little has been done to remove the requirement that all rays pass through a common point. We believe that the reluctance to remove the pinhole constraint is based on misconceptions.

One such misconception is that non-pinhole cameras (NPHCs) have no much use in graphics beyond automatically producing images that deviate from the rules of perspective in the interest of more poignant artistic expression. We believe that NPHC images

are likely to benefit the users of many applications as long as the camera model exhibits some amount of coherence, which we loosely define as the property of projecting nearby 3D points to nearby image plane locations. Another important use of NPHC images is in the context of image-based rendering (IBR), where they can provide powerful intermediate scene representations from which the desired image is produced faster and at higher quality than when the scene is processed at its full complexity. When the scene geometry is known, any image can be enhanced with depth, and therefore any image can be warped to the desired view by reprojecting its depth and color samples.

Another misconception is that rendering with an NPHC is inherently inefficient and therefore impractical in the context of interactive computer graphics applications. The feed-forward graphics pipeline has proven to be the best approach when efficiency is at a premium. The projection stage efficiently maps primitives to relevant pixels, which avoids considering pixel-primitive pairs that do not yield an intersection. It is true that most NPHC models developed so far do not offer efficient projection, therefore rendering requires ray tracing or a large number of feed-forward rendering passes. However, powerful NPHC models can be devised such that they offer fast projection, which enables efficient rendering with the feed-forward approach.

The goal of this paper is to introduce a novel paradigm for interactive computer graphics based on designing flexible yet efficient non-pinhole camera models. The remainder of the paper is organized as follows. Section 2 describes general principles for camera model design. Section 3 reviews prior non-pinhole camera models. We illustrate the paradigm of camera model design with three examples: the occlusion camera (Section 4), the sample-based camera (Section 5), and the graph camera (Section 6). We have previously presented the occlusion camera [Mei 2005, Popescu 2006a] and the sample-based camera [Popescu 2006b] in detail. They are briefly reviewed here from the stand point of camera model design. Section 7 discusses results and sketches directions for future work.

2 Camera model design

We propose to solve challenging interactive computer graphics problems by camera model design. Instead of using one of the few "off the shelf" cameras, the camera model should be designed to best fit the needs of the application, and it should be dynamically optimized according the scene or data set of interest. Camera model design proceeds in two major steps.

Step 1: define rays of interest. For some applications the rays of interest are obvious and can be computed automatically from data such as scene model and desired view. For other applications defining the relevant rays require the insight of expert developers or users of that particular application. To facilitate the task of defining the rays of interest we generalize the definition of a camera ray to allow for rays that are not a straight line. We define a camera ray as the locus of 3D points that project at a given image plane location.

Step 2: develop efficient projection. The output of the first step is already the equivalent of a camera model. The list of rays of interest defines a camera, which could be used to render the scene by ray tracing. Following the possibly curved rays out into the scene is not fundamentally more challenging than following conventional straight rays. However, such an approach is inefficient. The goal of the second step is to develop a fast projection method given the rays of interest, which enables efficient feed-forward rendering. Just like in the case of regular rays, projection in the context of general rays is defined as finding the image plane location(s) where a given 3D point is imaged, which implies finding the ray(s) that pass through the point. Depending on the complexity of the set of rays of interest, it could be impossible to develop an exact projection method that is also efficient. In such cases, an approximate projection method is developed. The approximation quality metric and the required quality depend on the application.

3 Prior work

Several NPHC models have been developed in the context of IBR for the purpose of scene modeling and rendering. The light field [Levoy 1996, Gortler 1996] is a 2D array of PPHC images. Light fields have the advantages of capturing a 3D scene while bypassing depth acquisition and of supporting complex rendering effects such as reflection and refraction at no extra cost. The light field camera model is powerful since it captures a dense set of rays, and it can be used in other graphics applications. An important limitation is the lengthy rendering time: the scene has to be rendered for each of the many PPHCs. Surface [Wood 2000] and unstructured [Buhler 2001] light fields reduce the number of rays using surface geometry, but construction remains an offline process. Light fields have to be used as a set of pre-computed color samples rather than as a set of rays, which precludes dynamic scenes.

Layered depth images (LDIs) [Shade 1998] generalize the PPHC image by allowing for more than one sample along a ray. The LDI camera can be thought of as a PPHC whose rays are broken into several segments. The application of LDIs is 3D image warping without the problem of disocclusion errors, which are artifacts due to surfaces visible in the desired view that are not sampled by the reference image. The number of samples stored at each LDI pixel varies widely; therefore it is impractical to render the LDI by successive rendering passes and by peeling off the nearest layer. Adequate LDIs are built by combining a large number of PPHC images rendered from views around the LDI reference view. Like in the case of light fields, LDIs are built offline and the LDI camera is too inefficient to accommodate dynamic scenes.

Multiple center of projection (MCOP) images [Rademacher 1998] collect samples with a vertical slit camera that slides along a user defined path. Possible goals in path selection are good scene coverage or artistic value of resulting image. The great flexibility in defining the rays of the MCOP camera makes it attractive in the context of our paradigm of model camera design. However, MCOP cameras are inefficient: images have to be rendered by ray tracing or by feed-forward rendering the scene for each center of projection along the camera path.

Another application of NPHC models is in the context of artistic rendering where they are employed to render multiperspective images. In one system individual PPHCs are attached to scene objects, and the resulting sprites are composited in a multi-projection image [Agrawala 2000]. For a small number of objects, the multi-projection image can be updated interactively. The approach of attaching a pinhole to each object has the disadvantages of not scaling with the scene complexity, of difficult—sometimes impossible—visibility ordering, and of not supporting multiple perspectives per object.

Another multiperspective rendering system [Yu 2004b] partitions an image plane into general linear camera (GLC) triangular images. A GLC is constructed from three given rays [Yu 2004b] so it offers the flexibility necessary for modeling a user specified set of rays. Another GLC advantage is fast projection

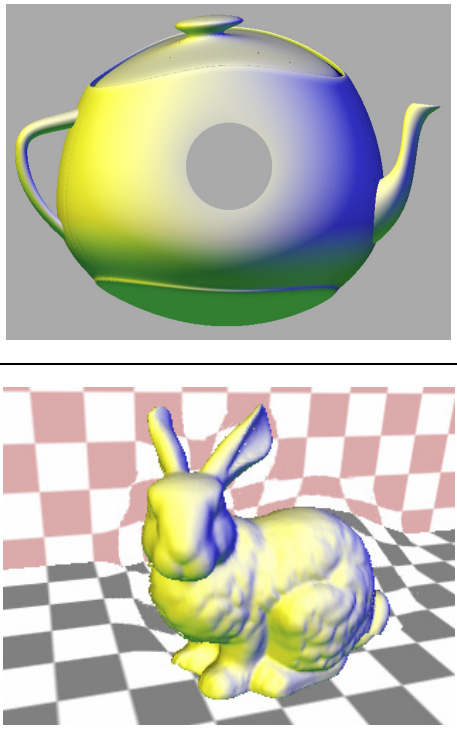

Figure 4 Single-pole (*top*) and depth-discontinuity (*bottom*) occlusion camera images.

[Popescu 2006c]. However, combining GLCs such that these advantages are preserved is non-trivial. The solution adopted by Yu et al. is to blend the rays of neighboring GLCs to provide a continuous ray space which generates an image with smoothly varying perspective. The resulting compound NPHC model does not provide fast projection and rendering is performed offline by ray tracing.

NPHCs have also been employed to facilitate the creation of panoramas for cel animation [Wood 1997]. The rays of interest are defined by the desired scene shots. The NPHC renders a multiperspective panorama which simulates camera motion in a 3D scene when it is viewed through a rectangular frame moving on a predetermined path. Similarly to the case of MCOPs, the panorama is rendered by finely discretizing the 3D camera path and by rendering an image for each position along the path. As it is the case for the other NPHC models discussed so far, using the multiperspective approach in the context of interactive computer graphics is hindered by the long rendering times.

In computer vision, designing NPHC models is complicated by several factors such as the mechanical and optical constraints of physical implementation and the unavailability of scene geometry. Multiperspective images are constructed by resampling a video cube—a stack of images gathered by moving a video camera along a continuous path [Seitz 2003]. The video cube has also been used to support impressionism, cubism, and abstract aesthetic video effects [Klein 2002]. Several simple NPHC models have been studied formally, such as the pushbroom camera [Gupta 1997], and the two-slit camera [Pajdla 2002], which are both part of the class of general linear cameras. Grossberg et al. describe a general camera model that defines rays explicitly [Grossberg 2001]. The model does not offer fast projection and using it in the context of computer graphics poses the problem of inefficient rendering, which has to be done by tracing individual rays.

Fast projection is also important in computer vision. Finding the image plane location of a given 3D point is a fundamental operation in common computer vision tasks such as camera calibration or correspondence search. Consider for example the application of depth from stereo using two NPHCs. Given a pixel in the left image, the locus of its possible correspondences in the right image can be found by projecting the left camera ray onto the right image. Fast projection would make epipolar-like constraints practical for NPHCs, and, in general, would help port the computational apparatus developed for pinholes to non-pinhole cameras.

4 The occlusion camera

The first example of camera model designed according to the proposed paradigm is the occlusion camera (OCC) [Mei 2005, Popescu 2006a]. The application of OCCs is to produce single-layer disocclusion-error-resistant reference images in the context of IBR. A reference depth image is asked to provide sufficient samples in order to reconstruct desired images from nearby views. The samples visible in the reference view do not suffice. The additional samples needed to prevent disocclusion errors are the samples that project close to the edge of occluders in the reference view, and are therefore "barely hidden".

Step 1. What is needed is a camera whose rays reach around occluders to gather some of the hidden samples—an occlusion camera. Such a camera is obtained by starting out with a reference PPHC and then by distorting its rays in 3D at the edges of occluders. The rays are bent in, towards the occluder.

Two approaches for specifying the distortion were implemented. The single-pole OCC is obtained by 3D distorting the rays radially around a pole. The pole is defined as the image plane projection of the centroid of the occluder (Figure 4, *top*). The single-pole OCC is suitable for simple scenes with one or a few major occluders. The depth-discontinuity OCC handles arbitrarily complex scenes by specifying the distortion independently for each ray using a distortion map (Figure 4, *right*). The distortion map is calculated from the depth discontinuity map: the distortion direction is perpendicular to the depth discontinuity direction and towards the occluder.

Step 2. At this step the details of the OCC model are determined such that the resulting camera offers fast projection. At step 1 the needed distortion has been described through its action on camera rays. The distortion has a similar effect on scene 3D points, except that the direction is reversed: bending the rays in, towards the occluder, is equivalent to pulling points out, away from the occluder. The point distortion magnitude should increase with z such that farther samples are distorted more. The goal is for hidden samples to clear the edge of the occluder and remain visible in the OCC image. A point is projected with an OCC by first projecting it with the corresponding undistorted PPHC and then by distorting it according to its depth.

Point projection is sufficient to render an image if a point-based reconstruction approach is taken. Higher-quality reconstructions require projecting triangles. PPHC rendering takes advantage of the fact that scene lines project to image plane lines. For an OCC straight lines project to curves and this complicates triangle rasterization. The simple distortion of the single-pole OCC allows computing a mapping from the OCC image plane to the triangle plane, which enables rasterization in the distorted domain. The

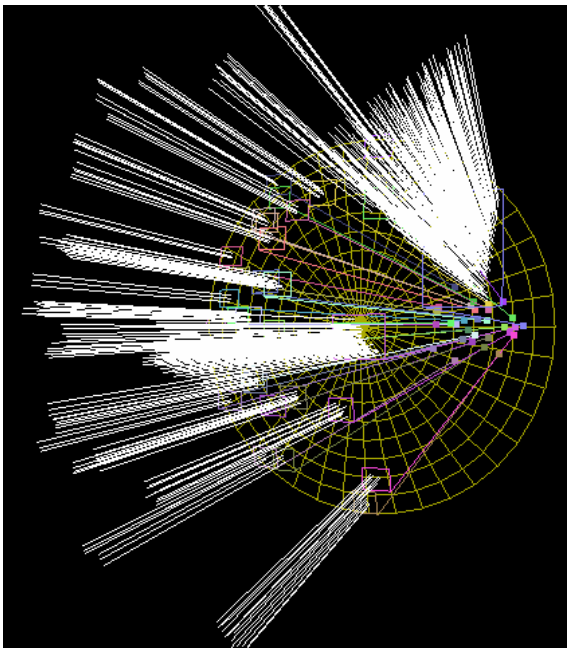

Figure 5 Visualization of a few simple cameras, together with the reflected rays they replace.

fine-grain controlled distortion of depth discontinuity OCC does not provide such a mapping, and the true curved edges of the projected triangle have to be approximated with lines. The approximation error is controlled by subdividing the scene triangles.

The resulting OCC produces a reference image that has sufficient samples for desired images from nearby views. The OCC image has a single layer and thus it shares advantages of regular (PPHC) depth images such as bounded number of samples, implicit connectivity, and efficient incremental processing. These fundamental advantages make the OCC a substantially better solution than LDIs, the previously best solution for disocclusion errors. The OCC was developed under the camera design paradigm. The camera model is tailored to the application and it is defined based on the scene.

5 The sample-based camera

The second example of camera model design is the sample-based camera (SBC) [Popescu 2006b]. The application of SBCs is rendering reflections. The approach of choice in interactive computer graphics is feed-forward rendering, with its two main stages of projection and rasterization. Unfortunately, reflections cannot be readily handled with this approach since one cannot easily find the projection of a reflected vertex. Reflected vertices can be easily projected if the reflector is planar, but for general reflectors there is no closed form solution to the projection equation. The SBC is a general camera that allows projecting vertices that are reflected in curved reflectors. The SBC model was designed as follows.

Step 1. The rays of interest in the case of reflections are obvious: they are the reflected rays generated by the desired view and the reflectors in the scene. However, the set of reflected ray can be large: the number of rays could be equal to the number of pixels if the entire screen is covered by reflectors. Although the rays leaving the eye are perturbed by the reflector, a considerable

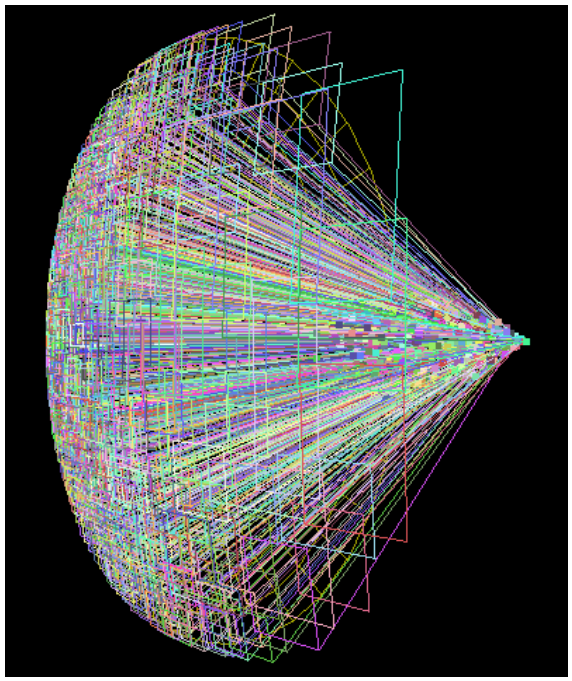

Figure 6 Visualization of sample-based camera comprising 728 simple cameras.

amount of coherence remains. The goal is to take advantage of this coherence in order to reduce the original set of reflected rays to a representative subset. The selected rays should model the original set well, possibly within an application specified approximation error tolerance. Moreover, ray selection should be fast since reflected rays depend on the desired view and ray selection has to run once per frame.

Ray selection begins by computing a reflected ray map that stores a reflected ray at each pixel covered by a reflector. The first-order ray map is computed in hardware by rasterizing the reflector triangle and by computing per-pixel reflected rays from vertex normals. The ray map is then subdivided in quadtree fashion until a rectangular region in the ray map can be approximated well by the four corner rays. The quality of the approximation is judged by fitting a simple camera to the four corner rays. The approximation error is given by the projection error of the simple camera, which is estimated at the center of the rectangular region. If the projection error is smaller than an application chosen limit, the recursive subdivision stops and the reflected rays are replaced with the simple camera (Figure 5). The simple cameras are modeled as PPHCs or as more powerful continuous 3-ray cameras, which are simple non-pinholes with closed form projection [Popescu 2006c].

Step 2. The algorithm described above quickly finds the rays of interest and uses them to build a set of simple cameras (Figure 6). However, depending on the desired projection accuracy, on the curvature of the reflector, and on the distance from the reflector to the desired viewpoint, the number of simple cameras can be large (Figure 7). For this it is inefficient to project every scene vertex in every simple camera. Projection is accelerated by arranging the simple cameras at the leafs of a binary space partitioning (BSP) tree. Given a vertex, the BSP tree is used to efficiently find the simple cameras with which the vertex needs to be projected.

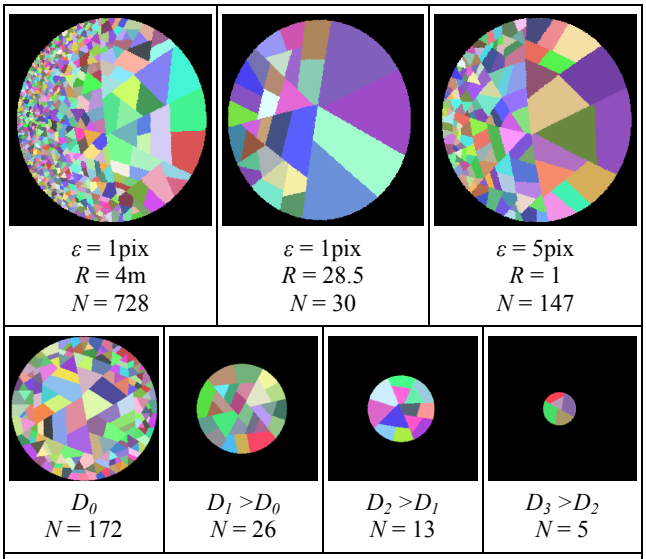

Figure 7 Visualization of reflector surface subdivision induced by simple cameras. The number of simple cameras N decreases with the increase of the projection error threshold ε, with the increase of the curvature radius R, and with the increase of the distance from the viewpoint to the reflector surface D.

In conclusion, the SBC is a compound camera designed for efficiently projecting reflected vertices. The SBC casts the problem of rendering reflections on curved objects in terms tractable by the feed-forward graphics pipeline. The SBC depends not only on the desired view, but also on the reflective geometry it encompasses.

6 The graph camera

A third example of camera model design is the graph camera, a novel camera that enables convenient visualization of heavily occluded environments. When a user explores a complex virtual 3D scene, navigation is challenging. Avoiding obstacles, inspecting hidden parts of the scene, and simply covering great distances in large-scale scenes reduce the efficiency with which the user assimilates information about the scene. Moreover, the interface through which the user has to specify the desired view is often non-intuitive.

For applications where the experience of actual locomotion in the virtual scene is unnecessary, user navigation can be reduced or even avoided by visualizing the scene with a comprehensive NPHC image that shows more of the scene than a regular PPHC image. The approach is supported by the high resolution of today's displays, which can show in detail such complex NPHC images. LCD's with over 9 million pixels are available for over four years now [IBM T221].

When compared to a set of PPHC images, each covering a different part of the scene, a comprehensive NPHC image has the advantage of visualization continuity. Although the rules of single perspective are distorted, an NPHC image allows the user to develop a better intuition for the overall scene than a set of disjoint PPHC images. Using overlapping PPHC images alleviates the problem somewhat but introduces costly redundancy.

Consider the example of a video surveillance application in the context of a building monitored with security cameras. The current approach is to display the images captured by the cameras on an array of monitors. First, there are typically fewer monitors

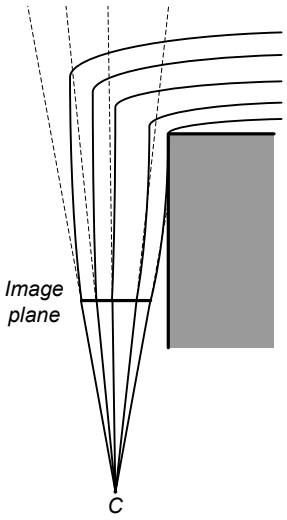

Figure 8 Ray distortion to enable "seeing" around the corner.

than cameras, which implies that several cameras share the same monitor. The time sharing follows a pre-determined algorithm or is orchestrated by the user (security guard), but either way, important security events can be missed. Second, inspecting the monitors reliably is challenging. The user has to scan the monitors sequentially. The effective time allotted to a monitor is further reduced by the need to adapt visually and cognitively to the image of each monitor. The application would benefit from a comprehensive view of the building obtained by integrating the multiple video feeds into a single NPHC image.

Another application example is in the visualization of large scale computer simulations. The raw performance of computing hardware and the sophistication of numerical codes have reached a stage that enables simulating with high temporal and spatial resolution the interaction of complex entities under extreme conditions. Visualization is an essential tool for designing, validating, and disseminating the results of such a simulation. Visualizing the dynamic scene with an NPHC image has the advantage of allowing the user to examine simultaneously several locations of interest in synchronized and visually coherent manner. For example, in our simulation of the September 11 Attack on the Pentagon, the interaction between the jet fuel—which concentrated most of the kinetic energy of the aircraft—and the numerous columns—the most relevant structural element of the building—was of highest interest to the civil engineering researchers [Popescu 2003, 2005]. An NPHC image could have shown simultaneously and in detail many of the columns affected.

6. 1 Design of graph camera model

Step 1. The rays of interest are defined by considering the parts of the scene that the user desires to visualize simultaneously. There have to be rays that sample each of these parts, and the image has to be as coherent as possible. A convenient way of specifying the rays of interest is to start with a PPHC and then to bend its rays making sure that they reach all the parts of the scene to be visualized. For example in Figure 8, the rays (see dotted lines) of a PPHC with center of projection C are bent to reach around the grey corner. The resulting NPHC model has one C^0-continuous ray per pixel. In addition to the literal malleability of the camera

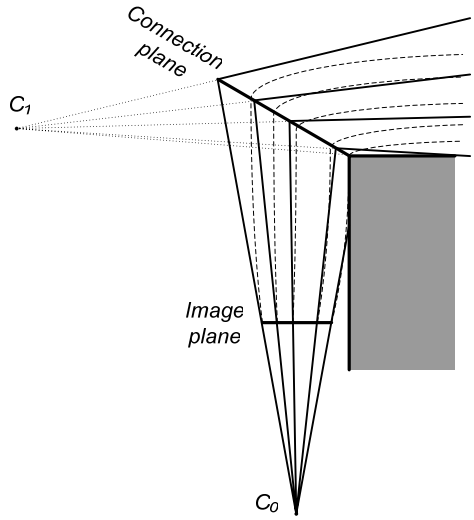

Figure 9 Seeing around the corner using two connected pinhole cameras C_0 and C_1.

model, this also brings the advantage of unambiguous visibility. A ray describes a rigorous order of the points that project at the respective pixel, which avoids visibility problems specific to other NPHC models [Agrawala 2000].

Step 2. One could simply define each individual ray as a set of adjoining curve segments and render by tracing each of the rays into the scene. In the interest of efficiency we model a general C^0 continuous ray as a chain of straight line segments, and we group the ray segments in a set of pinhole cameras. The piecewise linear approximation does not reduce the flexibility of the NPHC camera model significantly, and the set of pinholes allow rendering the NPHC image efficiently in hardware in feed-forward fashion. In Figure 9, the same effect as in Figure 8 is obtained using two pinhole cameras with centers of projection C_0 and C_1. The curved rays replaced by the two PPHCs are shown with dotted lines. The two pinhole cameras are connected at a plane.

Like in the case of sample-based cameras, the NPHC is decomposed in a set of simple cameras. The NPHC image is rendered by rendering the scene with each pinhole camera and compositing the individual images using a painter's style algorithm. Depth testing is unnecessary since the images are sorted in visibility order by construction. For the simple example in Figure 9, the image of C_1 should only overwrite the background pixels of the image gathered by C_0.

In order to implement the graph camera, several issues have to be addressed. First, the pinhole cameras used to model segments of the rays are not *planar* pinhole cameras. Even though a plane is used to describe the virtual imaging surface, the ray-image plane intersections do not define a regular 2D grid. We describe an efficient general pinhole camera (GPHC) model in Section 6. 2 . Second, the graph camera model has to be flexible, and we describe constructing GPHCs by folding the rays of a given GPHC in Section 6. 3 , by splitting a given GPHC in Section 6. 4 , and by splicing two adjacent GPHCs in Section 6. 5 .

6. 2 General pinhole camera

A simple GPHC model is obtained by enhancing a PPHC model with two scalars per pixel which indicate the actual image plane

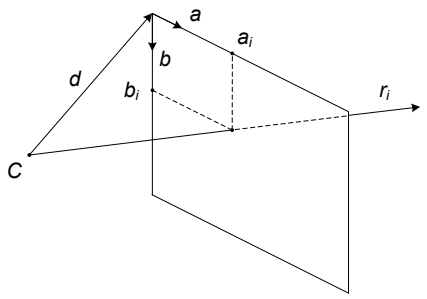

Figure 10 General pinhole camera. Ray r_i is defined by image plane coordinates (a_i, b_i).

point where the pixel ray intersects the image plane. In Figure 10, C marks the center of projection, and the vector d translates C to the origin of the image plane coordinate system with axes defined by vectors a and b. Each GPHC ray is explicitly defined by its intersection with the image plane. Like in the case of a PPHC, field of view limitations can be overcome by using a cube map centered at the center of projection.

Given a 3D point P one can trivially compute its projection by intersecting CP with the image plane. However, finding the closest pixel requires a search. Rendering a triangle with such a GPHC is even more expensive. While it is easy to compute the projected triangle by projecting each of the three vertices, it is difficult to find the pixels covered by the triangle.

For the GPHCs of interest to the application at hand, the rays sample the image plane fairly uniformly. This enables an efficient implementation of the GPHC based on defining an auxiliary PPHC whose resolution is fine enough such that there is at most one GPHC ray per PPHC pixel. The GPHC and the auxiliary PPHC have the same center of projection and the same image plane. The PPHC pixels are equal image plane parallelograms. If for a given PPHC pixel there is a GPHC ray that intersects the image plane within the pixel's parallelogram, then the intersection is encoded with a pair of floats that give the offsets from the top left corner of the parallelogram.

In Figure 11 the GPHC has 10 rays and it is modeled with an auxiliary PPHC with a 4x3 image plane. The two grey shaded PPHC pixels do not store a GPHC ray. We call such a pixel *inactive*. GPHC ray r_i is stored by PPHC pixel *(u, v)*. The intersection between r_i and the image plane is recorded with two offsets x_{uv} and y_{uv} and can be reconstructed with the expression $C+d+au+bv+ax_{uv}+by_{uv}$. The offset mechanism is similar to the

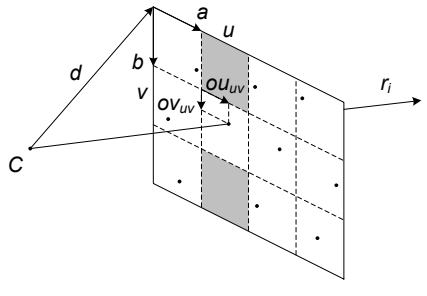

Figure 11 General pinhole camera implemented with an auxiliary planar pinhole camera.

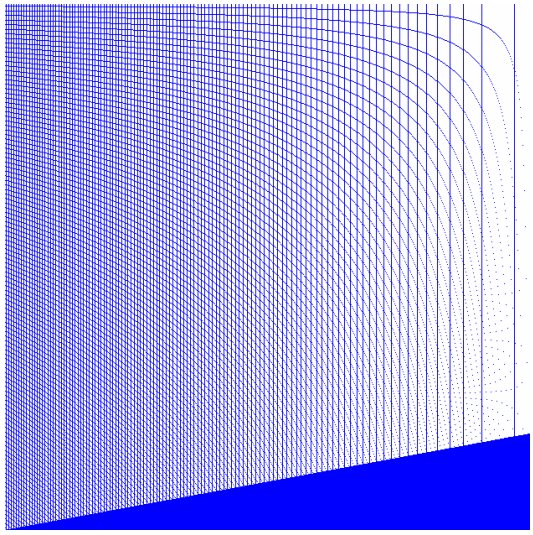

Figure 12 Visualization of the inactive auxiliary PPHC pixels, shown in blue.

one used in image-based rendering by warping to accurately locate the projection of a forward mapped depth and color sample [Popescu 2000].

When determining the resolution of the auxiliary PPHC, one has to make sure that there is no more than one GPHC ray per PPHC pixel. A possible solution is to find the smallest distance m between any two GPHC ray-image plane intersections and to use a PPHC with square pixels with a diagonal of m. In practice, this approach is too conservative: it generates a large number of inactive PPHC pixels. For the graph camera, the rays of the GPHCs are organized in a 2D mesh. Using Figure 9 again, the second pinhole is a GPHC that continues the rays of the first pinhole, which is a PPHC. The rays $(u+1, v)$, $(u, v+1)$, and $(u+1, v+1)$ which are neighbors of ray (u, v) for the first camera will also remain the neighbors of ray (u, v) for the second camera. We determine the resolution of the auxiliary PPHC for the second camera by computing the minimum distance between the image plane intersections of neighboring rays. Figure 12 shows the inactive pixels for the auxiliary PPHC of a typical GPHC. In this particular case there are 512x512 GPHC rays and the auxiliary PPHC image has a resolution of 627x627.

The auxiliary PPHC allows rendering triangles by projection followed by rasterization. Rasterization proceeds on the uniform pixel grid, which allows evaluating the bounding box of the projected triangle and the edge expressions at the current pixel. The inner most rasterization loop simply continues to the next iteration if the current pixel is inactive. For an active pixel, the rasterization parameters are computed at the actual GPHC ray by modulating the rasterization parameter values for the PPHC pixel using the pair of offsets. Let r be a rasterization parameter whose linear model space variation is expressed by the equations

$$r(u, v) = (A_r u+B_r v+C_r)/w(u, v)$$
$$w(u, v) = A_z u+B_z v+C_z$$

Examples of rasterization parameters include color channels (R, G, B, A), depth (z), normal components (n_x, n_y, n_z), texture coordinates (s, t), and any parameter desired by a custom shader. The value at the GPHC ray is given by $r(u+x(u, v), v+y(u, v))$, where $x(u, v)$ and $y(u, v)$ are the offsets at the current pixel.

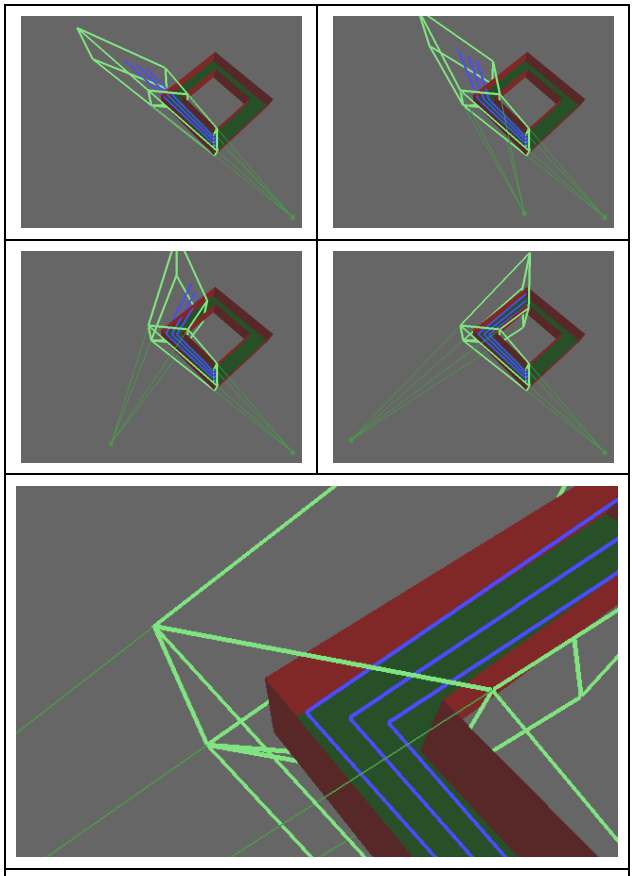

Figure 13 Visualization of folding operation.

The rasterization parameter modulation can be easily implemented in a shader that looks up the offsets in a texture. However, visibility will be decided along the center of the PPHC pixel since the shader runs after visibility and current programmable graphics hardware does not expose the visibility stage to the application. The approximation is only visible at edges where the actual GPHC ray might have an intersection with the foreground object whereas the ray through the center of the PPHC pixel might not have one, or vice versa.

True hardware support for the GPHC would consider the map of offsets as part of the camera model and would directly rasterize at the offset locations. In the mean time we use an efficient implementation that renders the scene with the auxiliary PPHC using the fixed pipeline and then resamples to form the GPHC image using the map of offsets. Bilinear interpolation produces good results since the sampling rates of the GPHC and of the PPHC are similar.

6. 3 Folding operation

The main operation needed to build a graph camera is folding the view frustum of a GPHC in order to change the perspective and see around corners or other occluders. Given a GPHC G_0 and a plane Π, the rays of G_0 are folded at Π with the following steps.

A. Construct GPHC G_1 to continue the rays of G_0 beyond Π.
   a. Construct auxiliary PPHC camera P_1
   b. Compute offsets and inactive pixels

B. Limit the view frustum of G_0 to Π.

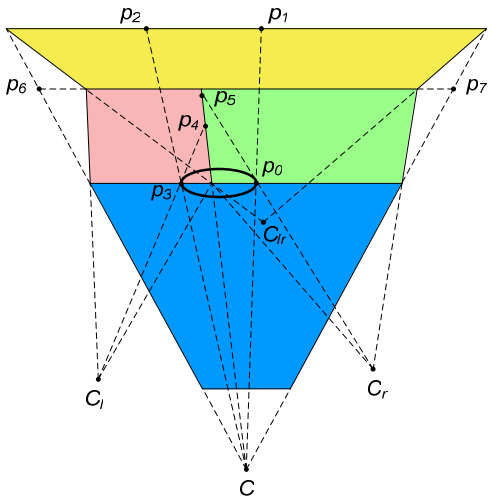

Figure 14 Illustration of splitting followed by splicing in order to see behind occluder. The resulting graph camera consists of 4 GPHCs, shown with different colors.

To construct the auxiliary camera P_1 the rays of G_0 are first intersected with Π. Then a 2D coordinate system is defined on Π, which is used to compute an axis aligned bounding box of the intersections. The bounding box defines the image frame of P_1. The resolution of P_1 is determined as described in the previous section. The center of projection of P_1 is chosen to define the desired change in perspective. Once P_1 is constructed, the intersections between the rays of G_0 and Π are projected on P_1 to set the offsets to the actual GPHC ray. P_1 pixels that do not receive any intersection point are marked as inactive. The rays of G_0 end at Π, which is implemented with a clipping plane that discards the half space on the opposite side of Π compared to the center of projection of G_0.

In Figure 13 the view frustum of a GPHC is folded 90° to the right. Intermediate positions are computed for illustrative purposes. Three rays are also shown. The rays are C^0-continuous at the connection plane. In Figure 3 the view frustum is folded 90° three times to effectively linearize the rectangular corridor. The resulting graph camera consists of 4 GPHCs, one for each corridor section.

6. 4 Splitting operation

Based on the folding operation, splitting a GPHC into two or more GPHCs is straight forward. Given a GPHC G_0 and a plane Π, G_0 is split into two GPHCs G_{0l} and G_{0r} by first partitioning the image plane of G_0 into two disjoint regions to create two GPHCs G_{0l}^* and G_{0r}^* and then by folding G_{0l}^* and G_{0r}^* as before to form G_{0l} and G_{0r}. G_{0l}^* and G_{0r}^* could have overlapping frusta, which implies that some scene parts can appear twice in the resulting graph camera image. A separator plane implemented as a clipping plane can be used to keep the two frusta disjoint.

6. 5 Splicing operation

The splicing operation is the reverse of splitting. Two GPHCs are spliced into a single GPHC at a plane Π by intersecting the rays of each GPHC with Π and then by unioning the intersections to form a single GPHC whose frustum begins at Π. Figure 14 shows how

Figure 15 Continuous morph as object traverses boundary between connected GPHCs.

a GPHC with center of projection C (*blue*) is first split into two GPHCs with centers of projection C_l (*red)* and C_r (*blue*), and then how the two GPHCs are spliced into GPHC C_{lr} (*yellow*). The resulting graph camera composed of 4 GPHCs sees behind the ellipsoidal occluder. The shadow of the occluder is reduced from the area $p_0p_1p_2p_3$ to the area $p_0p_5p_4p_3$.

The view frusta of C_l and C_r can be kept disjoint using a separator plane through p_4 and p_5. The separator plane clips some of the rays. For example the C_l ray through p_3 ends at the separator plane. This simply means that the graph camera ray Cp_3 ends at p_4. The maximum depth along the ray poly line is given by the distance $Cp_3 + p_3p_4$. Since some of the rays of C_l and C_r do not reach the plane defined by p_6p_7, camera C_{lr} has fewer rays than the first camera C. If a ray clipped by the separator plane does not encounter any surface before it ends at the separator plane, its corresponding pixel remains uninstantiated. The separator plane acts like a yon plane. An actual graph camera built according to Figure 14 is visualized in Figure 2, and the images it produces are shown in Figure 1. When the separator plane is used (Figure 1, *middle*), the bunny is not sufficiently large to occlude the separator plane hence the grey regions in the middle of the image. Not using the separator plane (Figure 1, *right*) covers all pixels at the cost of redundancy.

7 Discussion

The graph camera is a versatile yet efficient camera model that captures in a single image a complex 3D scene, reducing or even eliminating the need for navigation in the virtual environment. Multiple perspectives are integrated in a coherent, mostly continuous (Figure 15), visibility ordered (Figure 16) image.

The graph cameras used in this paper consist of 4 GPHCs. The scene's geometric complexity is given by the bunny statue which has ~70 thousand triangles. The scene is rendered once for each of the GPHCs (Figure 17) and the average frame rate is 5fps (3GB 3GHz Pentium 4 Xeon PC with an Quadro FX 3400 NVIDIA graphics card). The frame rate only increases to 5.5fps if the

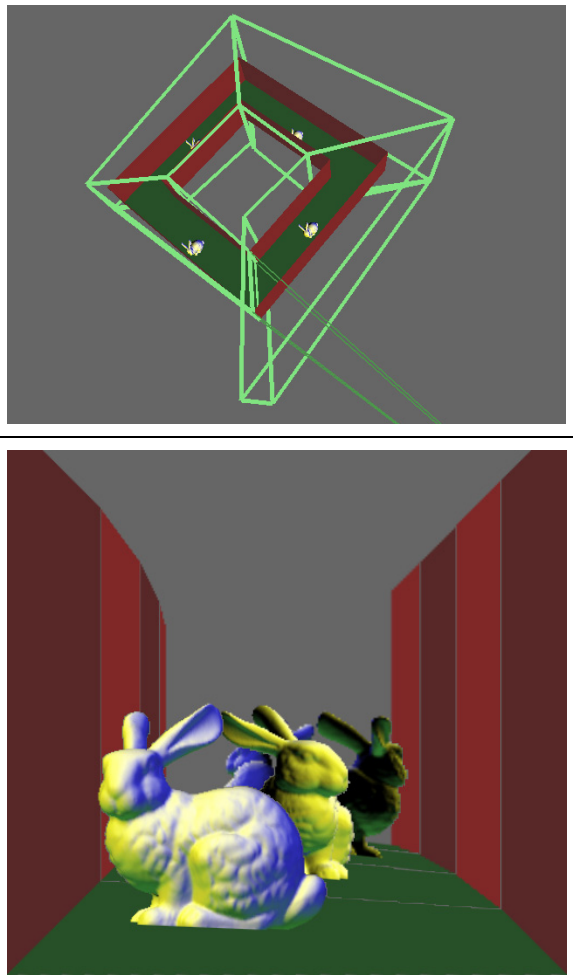

Figure 16 Illustration of correct visibility ordering. The four bunnies along the corridor (*top*) are correctly sorted near to far in the graph camera image.

bunny is not drawn, which indicates that the main factor affecting performance is reading back the 4 auxiliary PPHC images and resampling them to the graph camera image.

Performance can be increased by making better use of existing graphics hardware. We will investigate resampling on the GPU, which will also save the cost of reading the framebuffer back in main memory. Heavier geometry loads will warrant separating the view frusta of the GPHCs with a hierarchical space subdivision. As in the case of sample-based cameras, a BSP tree should provide a good solution. When the graph camera consists of a small number of GPHCs, the scene could be rendered in a single pass by providing the BSP tree to the vertex program via an array of uniform parameters. Special care is needed to handle the triangles that intersect more than one auxiliary camera frustum.

True hardware support for the model camera design paradigm is achieved if the graphics hardware is extended to offer programmability at the camera model level. The current programmability at vertex level is primarily intended for computing at each vertex the rasterization parameter values needed in the shader to evaluate the fragment color. Multiple projections, clipping with view volume walls that are not a plane but rather a higher-order surface, and rasterization of projected

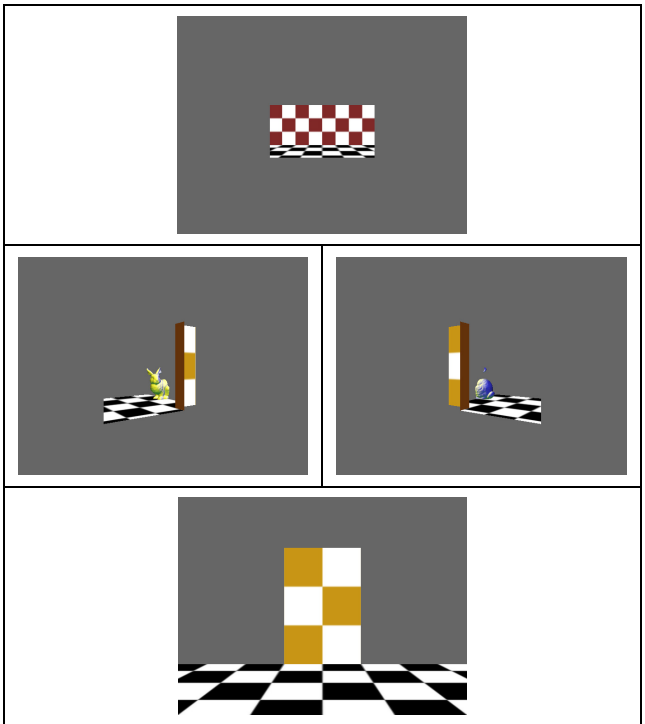

Figure 17 The 4 auxiliary PPHC images used to build the graph camera image in Figure 1—middle.

triangles with curved edges are not supported in hardware yet they are essential infrastructure for many NPHC cameras. For example, once the application developer specifies the projection function of the newly developed camera model, the hardware should be able to render a triangle with the new camera in feed forward fashion by subdividing the triangle as needed in order to meet an application specified error threshold for the approximation of the curved edges.

We will investigate how to implement the graph camera physically in order to provide support for applications such as video surveillance. Still and video digital cameras have become ubiquitous and they can be used in sufficient numbers to capture most of the needed rays. The challenges that have to be overcome include camera intrinsic and extrinsic calibration, camera placement in the context of physical constraints, and implementation of clipping planes. Although the precise geometry of the real-world environment is not known, a proxy of the background geometry should suffice for providing by 3D warping the rays that cannot be captured directly.

This paper discusses the occlusion camera and sample-based camera from the stand point of camera model design and introduces graph cameras. It is our hope that this work contributes substantially to the argument that non-pinhole cameras can be useful and efficient. Such NPHCs enable the model camera design paradigm, which opens the door to a new approach for finding solutions of difficult problems in computer graphics and beyond.

8 Acknowledgments

We would like to thank Chunhui Mei, Jordan Dauble, Elisha Sacks, Dan Aliaga, Chris Hoffmann, and the entire computer science graphics group at Purdue University for useful discussions. This work was supported by NSF grant SCI-

0417458. Equipment was donated by Intel and IBM. The bunny geometric model is courtesy of the Stanford 3D Scanning Repository.

References

[Agrawala 2000] M. Agrawala, D. Zorin, and T. Munzner. Artistic multiprojection rendering. In *Eurographics Rendering Workshop 2000* (2000).

[Buhler 1999] C. Buhler et al. Unstructured Lumigraph Rendering. *Proc. SIGGRAPH 2001,* (2001).

[Chang 1999] C-F Chang, G. Bishop, and A. Lastra. LDI Tree: A Hierarchical Representation for Image-Based Rendering, *Proc. SIGGRAPH'99*, (1999).

[Gortler 1996] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen. The Lumigraph. *Proc. of SIGGRAPH* 96, 43-54.

[Grossberg 2001] D. Grossberg and S. Nayar. A General Imaging Model and a Method for Finding its Parameters. In *Proceedings of ICCV 2001* (2001).

[Gupta 1997] R. Gupta, R. I. Hartley. Linear Pushbroom Cameras. *IEEE Trans. Pattern Analysis and Machine Intell.* vol. 19, no. 9 (1997) 963–975.

[IBM T221] http://www.ibm.com/us/

[Klein 2002] A. Klein, P.-P. Sloan, A. Finkelstein, M. F. Cohen. Stylized Video Cubes. *Proc of Symposium on Computer Animation 2002,* 15-22 (2002).

[Levoy 1996] M. Levoy, and P. Hanrahan. Light Field Rendering. *Proc. of SIGGRAPH* 96, 31-42 (1996).

[Mei 2005] C. Mei, V. Popescu, and E. Sacks. *The Occlusion Camera*. In proc. of Eurographics 2005, Computer Graphics Forum, vol. 24, issue 3, sept 2005.

[McMillan 1995] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *In Proc. SIGGRAPH* '95, pages 39-46, 1995.

[Pajdla 2002] T. Pajdla. Geometry of Two-Slit Camera. *Research Report CTU–CMP–2002–02*, 2002.

[Popescu 2006a] V. Popescu and D. Aliaga. The Depth Discontinuity Occlusion Camera. In Proc. of ACM Symposium on Interactive 3D Graphics and Games, 2006.

[Popescu 2006b] Popescu, V., E. Sacks, and C. Mei. Sample-Based Cameras for Feed-Forward Reflection Rendering. To appear in IEEE Transactions on Visualization and Computer Graphics, (2006).

[Popescu 2006c] Popescu, V., C. Mei, J. Dauble, and E. Sacks. An Efficient Error-Bounded General Camera Model. In *Proceedings of 3rd International Symposium on Data Processing, Visualization, and Transmission* 2006.

[Popescu 2005] Popescu, V., and C. Hoffmann. Fidelity in Visualizing Large-Scale Simulations. Journal of Computer Aided Design, Elsevier, 37 (2005), pp 99-107.

[Popescu 2003] Popescu, V., C. Hoffmann, S. Kilic, M. Sozen, S. Meador. Producing High-Quality Visualization of Large-Scale Simulations. In *Proceedings of IEEE Visualization*, (2003).

[Popescu 2000] Popescu, V., et al. The WarpEngine: an Architecture for the Post-Polygonal Age. In *Proc of SIGGRAPH'00*, pp 433-442, (2000).

[Rademacher 1998] P. Rademacher, G. Bishop. Multiple-center-of-Projection Images. *Proc. ACM SIGGRAPH* '98 (1998)199–206.

[Seitz 2003] S.M. Seitz, J. Kim. Multiperspective imaging. *Computer Graphics and Applications*, IEEE Volume 23, Issue 6, Nov.-Dec. 2003 Page(s):16 – 19.

[Shade 1998] J. Shade, S. Gortler, L. He, et al. Layered Depth Images, *In Proceedings of SIGGRAPH 98*, 231-242.

[Wood 1997] D. N. Wood, A. Finkelstein, J. F. Hughes, et al. Multiperspective Panoramas for Cel Animation. *Proc. ACM SIGGRAPH* '97 (1997) 243-250.

[Wood 2000] D. Woord, D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. Salesin, W. Stuetzle. Surface Light Fields for 3D Photography. Proceedings of SIGGRAPH 2000.

[Yu 2004a] J. Yu, and L. McMillan. General Linear Cameras *In 8th European Conference on Computer Vision (ECCV),* 2004, Volume 2, 14-27.

[Yu 2004b] J. Yu, and L. McMillan. A Framework for Multiperspective Rendering. *In Proceedings of Eurographics Symposium on Rendering (EGSR),* 2004.